Joakim Isaksson

Puzzlepelin suunnittelu ja toteutus Androidille

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
12.5.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Joakim Isaksson Puzzlepelin suunnittelu ja toteutus Androidille 32 sivua + 0 liitettä 12.05.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Jorma Räty Lehtori Tapani Martti

Työn tavoitteena oli suunnitella ja kehittää mobiilipeli androidille. Tarkoituksena oli perehtyä pelikehityksen eri osa-alueisiin käytännön toteutuksen kautta, mikä tehtiin osana opinnäytetyötä. Työssä seurataan pelin kehitystä aina ensimmäisestä ideasta lopulliseen ja hiottuun julkaisuversioon asti.

Työssä etsitään vastauksia useisiin pelisuunnittelua kiinnostaviin kysymyksiin, kuten mitä pelit ovat, miksi pelaamme ja mikä tekee pelistä hyvän. Pelin suunnitteluun ja testaamiseen keskitytään hieman kattavammin, kun taas teknisessä osuudessa pysytellään lähinnä arkkitehtuurissa, eikä kooditasolle juurikaan mennä.

Pelin tekeminen aloitettiin opiskelijan omasta harrastuneisuudesta ja kiinnostuksesta pelialaa kohtaan. Projektin käynnistyttyä työlle tarjoitui kuitenkin nopeasti asiakas, joka toi mukanaan pelille Kalevala-aiheisen teeman. Peli on nyt myyty Intiaan ja julkaistu useilla eri kielillä. Peli löytyy myös Samsung Storesta ja Google Playsta nimellä ”Sampo Lock”.

Avainsanat	Andoird, Pelisuunnittelu, Puzzle, Peli

Metropolia

Author(s) Title Number of Pages Date	Joakim Isaksson Designing and creating a puzzle game for Android 32 pages + 0 appendices 12 May 2014
Degree	Bachelor of Engineering
Degree Programme	Information Engineering
Specialisation option	Software Engineering
Instructor(s)	Jorma Räty, Senior Lecturer Tapani Martti, Senior Lecturer

The goal of the study was to design and develop a mobile game for Android. The aim was also to become more familiar with all the various areas of game development by implementing a game as part of the study. The paper presents all the development steps from the first idea to the final release.

The study seeks to answer a number of interesting questions about game design , such as what games are, why we play and what makes a good game. The study is focused more on the game desing and testing than the practical implementation of the game.

The development of the game began as a hobby, but quickly after the development had started a customer became involved in the project. The customer brought a new theme to the game based on the Finnish folk lore Kalevala. The game has already been sold to India and published in several different languages. The game can be found in Samsung Store and Google Play with the name " Sampo Lock".

Keywords	Android, Game designing, Puzzle, Game

Sisällys

Lyhenteet

2D, 3D	Two Dimensional, Three Dimensional. Kaksi- tai kolmiulotteinen järjestelmä.
API	Application Programming Interface. Ohjelmointirajapinta jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoa keskenään.
APK	Application Package File. Androidin käyttämä ohjelmatiedostojen pakkaustiedosto.
D*	D-Star. Inkrementaalinen hakualgoritmi, jota käytetään reittien etsimiseen tuntemattomassa ympäristössä.
IP	Intellectual Property. Immateriaalioikeus, joka voidaan jakaa tekijänoikeuksiin ja teollisoikeuksiin.
JSON	JavaScript Object Notation. Avoimen standardin tiedostomuoto tiedonvälitykseen.
LUA	Kevyt ja yksinkertainen skriptikieli, joka on tarkoitettu lähinnä ohjelmien helppoon laajentamiseen.
MVC	Model View Controller. Ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on käyttöliittymän erottaminen muusta sovelluksesta.
SDK	Software Development Kit. Kokoelma ohjelmistokehitystyökaluja.
Tile	Tile, eli laatta. Tekniikka 2D-grafiikoiden rakentamiseen pienistä neliön muotoisista kuvista.
XML	Extensible Markup Language. Helppolukuinen tekstipohjainen merkintäkieli.

1 Johdanto

Insinöörityön tavoitteena on suunnitella ja kehittää uusi mobiilipeli androidille. Työn tarkoituksena on tutustua pelikehityksen eri vaiheisiin ja rakentaa parempaa käsitystä pelikehityksestä kokonaisuutena. Aihetta lähdetään lähestymään käytännön toteutuksen kautta, joka tehdään osana opinnäytetyötä. Kehitettävän pelin avulla pelikehitystä on tarkoitus seurata aina idean synnystä lopulliseen versioon asti.

Työssä lähdetään liikkeelle etsimällä vastauksia pelisuunnittelun kannalta mielenkiintoisiin kysymyksiin, kuten mitä pelit ovat, miksi pelaamme ja mikä tekee pelistä hyvän. Työssä pyritään keskittymään pelin suunnitteluun ja testaamiseen, eikä pelin tekniseen toteutukseen syvennytä kovin tarkasti. Teknisessä osuudessa on tarkoituksena keskittyä pelin arkkitehtuuriin, eikä tarkastelussa aiota mennä kooditasolle asti.

Opinnäytetyön aihe valittiin kirjoittajan omasta kiinnostuksesta pelialaa kohtaan ja halusta oppia ymmärtämään pelikehitykseen kuuluvia prosesseja laajemmin. Työn toivotaan avaavan uusia näkökulmia pelikehitykseen ja avartavan nykyisiä käsityksiä pelikehitystä kohtaan.

2 Pelin ymmärtäminen

2.1 Pelit käsitteenä

Peli on käsitteenä hyvin laaja, eikä sille ole olemassa virallista määritelmää. Meillä kaikilla tuntuu olevan oma näkemyksemme siitä, mitä pelit ovat ja mitä ne tarkoittavat. Emme kuitenkaan törmää usein ristiriitaisuuksiin ja ymmärrämme yleensä hyvin, mitä tarkoitamme, kun puhumme peleistä kulloisessakin asiayhteydessä. Pelisuunnittelun kannalta on kuitenkin hyödyllistä, että pohdimme asiaa hieman syvällisemmin. Jokaisella pelisuunnittelijalla on oma määritelmänsä pelille, mutta tärkeintä ei ole oikean lopputuloksen löytäminen vaan sen etsiminen. [1.]

Aloitetaan pelin määritteleminen olettamuksella: ”Peli on jotain mitä voidaan pelata”. Tämä ei kuitenkaan kerro meille vielä kovinkaan paljoa, koska samaa voidaan sanoa esimerkiksi leluista ”lelu on jotain millä voidaan leikkiä”. Lelut eivät kuitenkaan ole pelejä, eikä leikkiminen ole pelaamista, mutta jotain yhteistä näillä kuitenkin tuntuu olevan. Lelut vaikuttavat olevan pelejä yksinkertaisempia, joten voimme yrittää ensin määritellä, mitä lelut oikein ovat. Voimme leikkiä leluilla, mutta voimme leikkiä myös ystäviemme kanssa, jotka eivät kuitenkaan ole leluja. Lelut ovat siis esineitä, jotka on varta vasten tehty leikkimistä varten. Pelit eivät kuitenkaan ole aina fyysisiä esineitä, mutta ne on myös tehty aina erityisesti pelaamista varten. Mitä leikkiminen sitten oikeastaan on?

Käyttäytymistieteet ovat tutkineet leikkimistä ja sen vaikutuksia jo pitkään. Mielenkiintoista on, että leikkiminen ei rajoitu pelkästään ihmisiin vaan sitä esiintyy myös muissa eläinlajeissa. Leikkiminen on hyvin tärkeää lasten kehityksen kannalta, ja se on lapselle luontainen tapa oppia. Leikkimisen avulla lapsi tutustuu ympäristöönsä ja opettelee uusia asioita. Mitä tämä kertoo meille pelaamisesta ja kuinka se eroaa leikkimisestä? [3.]

Yksi asia, joka erottaa pelaamisen ja leikkimisen toisistaan, ovat säännöt. Peleissä ja leikeissä molemmissa on sääntöjä, mutta ne tulevat eri paikoissa. Leikin säännöt syntyvät leikin sisällä ja ovat leikkijöiden itsensä kehittämiä, kun taas pelien säännöt tulevat pelin ulkopuolelta ja ovat ennalta sovittuja. Lisäksi leikki loppuu yleensä vasta, kun leikkijät haluavat tai joutuvat lopettamaan. Peleillä on sen sijaan olemassa jokin voittoehto tai päämäärä, jota pelaajat tavoittelevat.

Olemme vasta päässeet raapaisemaan kysymysten pintaa, mutta voimme yrittää rakentaa lyhyttä määritelmää pelille asioista, joita olemme havainneet:

- Pelaaminen vaatii vähintään yhden pelaajan.
- Pelissä on ennalta sovittuja sääntöjä.
- Pelissä on päämäärä, tavoite tai voittoehto.

Testataan määritelmäämme kahdella eri tavalla. Ensiksi voimme varmistua, että määritelmämme ei ole liian geneerinen. Se ei saa päteä muihin asioihin, kuten esimerkiksi leluihin tai kirjoihin. Lelut eivät aseta tavoitteita eivätkä anna sääntöjä, mutta asettaako kirja lukijalleen sääntöjä? Kirjat noudattavat useita sääntöjä kieliopista

kirjan rakenteeseen. Tämä ei kuitenkaan tarkoita sitä, että kirjassa itsessään olisi sääntöjä kuten pelissä.

Toiseksi voimme yrittää keksiä pelejä, jotka eivät noudata määritelmäämme. Suurin osa peleistä tuntuu noudattavan määritelmäämme hyvin, mutta hankaliakin tapauksia löytyy. Erityisen ongelmallisia ovat simulaattoripelit, jotka eivät aseta pelaajille selkeitä tavoitteita. Suurin osa simulaattoripeleistä kuitenkin asettaa jonkinlaisia tavoitteita pelaajilleen, mutta tavoitteet eivät välttämättä ole selkeästi esillä. Varsinaiseksi ongelmaksi muodostuvat kuitenkin pelit, jotka eivät edes tunnu vaativan varsinaista pelaajaa. Brittiläisen matemaatikon John Horton Conwayn vuonna 1970 kehittämä peli ”Game of Life” ei ensisilmäyksellä näytä vaativan yhtään pelaajaa. Vaikka ”Game of Life” mielletään yleensä peliksi, niin on kyseessä kuitenkin vain digitaalinen versio lelusta, jolla käyttäjä voi leikkiä simuloimalla erilaisia tilanteita. [4.]

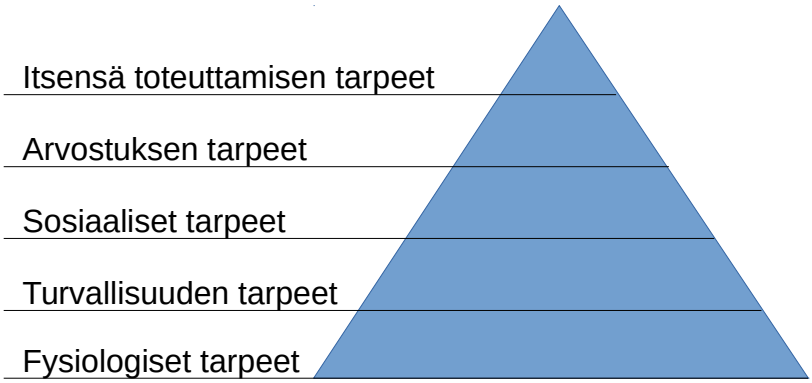
Määritelmämme pelille ei siis missään nimessä ole täydellinen, mutta opimme hiukan jotain pelien syvimmistä rakenteista. Voimme käyttää oppimaamme ymmärtääksemme paremmin pelisuunnittelun periaatteita.

2.2 Pelit tarpeina

Mikäli haluamme suunnitella menestyvän pelin, tulee meidän ensin ymmärtää, mikä saa pelaajat pelaamaan peliämme. Pelaajat haluavat hyviä pelejä, mutta mikä tekee pelistä hyvän? Tuhannet asiat vaikuttavat hyvän pelin lopputulokseen, eikä kaiken kattavaa vastausta ole olemassa. Voimme kuitenkin lähestyä kysymystä esittämällä seuraavan johdonmukaisen kysymyksen: Miksi pelaamme?

Pelaamme, koska pelaaminen on hauskaa, mutta mikä on hauska? Mikä vain voi olla hauskaa, koska hauskuus on täysin subjektiivista. Se mikä on jollekkin hauskaa, ei välttämättä ole toiselle hauskaa. Hauskat asiat tuottavat meille mielihyvää, mutta niin tuottaa myös syöminen. Emme kuitenkaan miellä syömistä hauskaksi, joten mitä muuta hauskuus tarkoittaa? Vitsit ovat hauskoja ensimmäisellä kerralla, mutta sama vitsi ei enää ole kovin hauska kolmannella kerralla. Ennalta arvaamattomuus näyttää siis liittyvän hauskuuteen. Ruokien maisteleminen sokkona voi olla hauskaa, kuten myös uusien asioiden kokeileminen. Hauskuus on siis vain mielihyvää yllätyksillä. Tiedämme mitä yllätykset ovat, mutta mikä tuottaa meille mielihyvää? [5.]

Pelaaminen ja leikkiminen täyttävät ihmisten monia psykologisia tarpeita, joista pelaaja ei yleensä itse ole edes tietoinen. Näitä evoluution myötä kehittyneitä tarpeita tutkii tarveteoria, joista kuuluisimpia ovat Maslowin tarvehierarkia ja Herzbergin motivaatioteoria. Maslowin tarvehierarkia jakaa tarpeet viiteen hierarkiseen luokkaan, joista ensimmäinen on perustavin ja viimeinen korkein tarve. Maslowin tarvehierarkia esitetään usein pyramidina (Kuva 1), jossa pohjimmaisina ovat perustarpeet ja huipulla korkeammat tarpeet. [6.]



Kuva 1. Maslowin tarvehierarkia

Mitä tarpeita pelaaminen sitten täyttää? Pelaaminen on yksi harvoista asioista, joka voi täyttää tarpeita kaikista kolmesta korkeimmasta luokasta. Erityisen koukuttavaa pelaamisesta tekee se, että tarpeet täyttyvät pelaamalla paljon nopeammin ja helpommin kuin normaalissa elämässä. Kun kysymme pelaajilta, miksi he pelaavat, niin yleisimmät vastaukset ovat:

- Pidän haasteista, pidän ongelmien ratkomisesta.
- Rentoutuakseni, irtautuakseni arjesta, karatakseni todellisuutta.
- Koska kaverini pelaavat, viettääkseni aikaa ystävieni kanssa. [1.]

Aluksi saattaa tuntua oudolta, miksi pidämme ongelmien ratkomisesta, koska ongelmat mielletään yleensä negatiivisiksi asioiksi. Evoluutio on kuitenkin muokannut meistä todellisia ongelmanratkontakoneita, mikä on yksi hyvän menestyksemme salaisuus. On siis täysin luonnollista, että ongelmien ratkominen tuottaa meille runsaasti mielihyvää. Maslowin tarvehierarkian mukaan ongelmien ratkaiseminen kuuluu itsensä toteuttamisen tarpeisiin. [7.]

Rentoutuminen, arjesta irtautuminen ja todellisuudesta karkaaminen eivät ole tarpeita itsessään vaan niillä kuvataan yleensä useiden eri tarpeiden täyttymistä. Pelimaailma

on todellisuutta yksinkertaisempi ja se on helpommin kontrolloitavissa. Pelit täyttävät pelaajien tarvetta hallita ympäristöään ja omia tekojaan. Yhdessä pelaaminen puolestaan tyydyttää pelaajien sosiaalisia ja arvostuksen tarpeita. Pelit ovat usein kilpailu- ja yhteistyöhenkisiä, jotka tarjoavat erityisesti nuorille pelaajille tärkeää yhteenkuuluvuuden tunnetta.

3 Pelin suunnittelu

3.1 Kokemusten suunnittelu

Pelisuunnittelua ajatellaan usein myös kokemusten suunnitteluna, koska pohjimmiltaan pelisuunnittelijat haluavat vain suunnitella hienoja kokemuksia. Pelit eivät kuitenkaan itsessään ole kokemuksia, mutta silti kokemukset ovat osa pelaamista. Kokemukset ovat eräänlaisia pelin ja pelaajan välisen vuorovaikutuksen sivutuotteita.

Kokemukset ovat subjektiivisia tuotteita, ja ne syntyvät pelaajassa itsessään. Emme pääse kokemuksiin suoraan käsiksi emmekä voi muokata niitä suoraan. Voimme kuitenkin suunnitella tietynlaisen pelin, jota pelaamalla pelaajalle kehittyy halutun kaltaisia tunteita ja kokemuksia. [1.]

3.2 Vaatimusten suunnittelu

Mobiilipelejä pelataan eniten kodeissa, kulkuneuvoissa ja käymälöissä, mutta vaikka pelejä pelataan hyvin vaihtelevissa ympäristöissä niin niiden tilanteeseen liittyy yleensä aina yksi ja sama asia - odottaminen. Mobiilipelaajat pelaavatkin yleensä lyhyitä sessioita kerrallaan, mutta useita kertoja päivässä. Vaikka tablettien yleistettyä myös mobiilipelaaminen kotisohvalla on yleistynyt, niin yksittäiset pelisessiot ovat pysyneet lyhyinä. Erittäin menestyneet mobiilipelit kuten Clash of Clans ovat hyödyntäneet näitä pelaamistottumuksia ja suunnitelleet koko pelin näiden ympärille. Clash of Clans kannustaa pelaajia pelaamaan vähän kerrallaan esimerkiksi asettamalla pelaajien tekemille toiminnoille odotusaikoja. [20.]

Puzzlepelit soveltuvat kuitenkin luonnostaan hyvin mobiilipeleiksi, koska ne eivät yleensä vaadi pitkää keskittymistä kerrallaan. Pulmapelit ovat yleensä helppoja

aloittaa; ne voidaan jättää kesken ja niitä on helppo jatkaa halutessa. Ei siis ole sattumaa, että merkittävä osa markkinoilla olevista mobiilipeleistä on jonkinlaisia puzzlepelejä. Yksinkertaisiin fysiikoihin perustuvat puzzlet ovat olleet erityisen kovassa suosiossa viime vuosina.

Mobiilipelaajat eivät siis halua omistaa paljon aikaa pelaamiselle kerrallaan, jonka lisäksi pelejä pelataan usein liikenteessä ja julkisissa paikoissa. Näiden tietojen pohjalta voidaan listata erilaisia vaatimuksia suunniteltavalle pelille:

- Yksittäinen pelisessio ei saa kestää liian kauan.
- Pelin aloittaminen pitää olla nopeaa ja suoraviivaista.
- Peli pitää voida jättää nopeasti kesken missä vain tilanteessa ja sen jatkaminen myöhemmin pitää onnistua helposti.
- Kosketusnäytön käyttö pitää olla helppoa, jopa kävellessä. Peli ei saa vaatia liikaa tarkkuutta kosketusnäytön käytön kanssa.
- Peli ei saa olla riippuvainen kiihtyvyys- tai kaltevuussensoreista.
- Äänet eivät saa olla merkitsevässä osassa ja ne tulee voida kytkeä pois päältä helposti missä vain kohtaa peliä.
- Hetkellisestä keskittymisen herpaantumisesta ei saa rankaista pelaajaa liikaa. Peli ei saa pohjautua mekaniikkoihin, jotka käyttävät tiukkoja aikarajoja tai vaativat nopeaa reagointikykyä.
- Pelissä pitää olla sisältöä pitkäksi aikaa ja uuden sisällön luominen pitää olla nopeaa ja helppoa.
- Pelillä ei saa olla tarvetta jatkuvaan ylläpitoon.
- Pelin pitää olla yhden miehen tehtävissä kohtuullisessa ajassa.

3.3 Idean sunnittelu

Mistä hyvät peli-ideat tulevat? Yleisin tapa saada peli-ideoita on inspiroitua jostain. Inspiraatiota voi hakea melkeinpä mistä vain, mutta hyvää ideaa on usein vaikea erottaa huonosta ideasta. Peli-ideoista ei kuitenkaan usein ole pulaa, eikä pelin alkuperäinen idea vaikuta lopullisen pelin menestykseen yleensä kovinkaan paljoa. Ideat ovat eräänlaisia lähtökohtia, jonka pohjalta peliä voidaan lähteä suunnittelemaan ja kehittämään järkeväksi kokonaisuudeksi.

Menestyvien ideoiden ei edes tarvitse olla alkuperäisiä, mutta yleensä pyrimme keksimään sellaisia. On täysin mahdollista, että alkuperäisiä ideoita on vielä olemassa jossain ideoiden ulottuvuudessa, mutta käytännössä alkuperäisiä ideoita ei ole olemassa. Kaikki keksimämme pohjautuvat jollakin tavoin jo olemassa olevaan tietoon, joka on täysin luonnollista. Aivomme eivät yksinkertaisesti kykene ajattelemaan asioita, jotka eivät pohjautuisi mihinkään aikaisempaan ajatukseen. Uudet ja alkuperäiset ideat ovat siis yleensä vai hieman muunneltuja vanhoja ideoita.

Pelistä hyvän tekee sen suunnittelu ja toteutus eikä itse idea. Jokainen peli tarvitsee kuitenkin idean, joka määrittää pelille suunnan ja toimii eräänlaisena ponnahduslautana. Huonoja ideoita ei siis ole olemassa - ainoastaan huonoja toteutuksia. Lopputuloksen kannalta on tärkeää, että myös idean annetaan elää ja kehittyä koko pelin kehityksen ajan. Monet ideat saattavat kuulostaa oudoilta paperilla, mutta osoittautuvat hyviksi käytännössä. Esimerkkejä menestyneistä, mutta oudoista ideoista:

- Tasohyppely, jossa italialainen putkimies, joka saa supervoimia syömällä sieniä yrittää pelastaa prinsessan kilpikonnien kynsistä. - Super Mario
- Tasohyppelyn ja flipperin yhdistelmä, jossa sininen siili pyrkii vapauttamaan hullun professorin vangitsemia eläimiä. - Sonic
- Leegot digitaalisessa muodossa, jossa pelaajat voivat kaivaa ja rakentaa neliöistä koostuvaa pelimaailmaa vapaasti ilman pelin asettamia tavoitteita. - Mine Craft

Sampo Lockin peli-idea syntyi ideasta luoda yksinkertainen ”Fifteen Puzzle”:n kaltainen peli, jossa pelaajan on liikuteltava neliön muotoisia paloja kaksiulotteisessa ahtaassa tilassa saadakseen palat oikeaan järjestykseen. Numeroiden järjestäminen on kuitenkin tavoitteena tylsä ja siitä haluttiin tehdä mielenkiintoisempi.

Kuva 2. Ratkaistu Fifteen Puzzle

Mitä jos lopullinen ratkaisu ei olisi ennalta tiedossa, kuten esimerkiksi vanhoissa pulmapeleissä. Pelaajan haluttiin etsivän oikeaa ratkaisua, eikä ainoastaan tapaa päästä siihen. Tarvittiin uusi tapa yhdistää palat toisiinsa.

Ensimmäisenä ajatuksena oli käyttää numeroiden sijasta kuvaa, mutta normaali kuva voi olla ennalta arvattavissa tai tiedossa. Seuraava idea oli käyttää abstraktia kuvaa, joka muodostuisi esimerkiksi yhdestä katkeamattomasta viivasta. Tämä ei kuitenkaan vaikuttanut tarpeeksi mielenkiintoiselta visuaalisesti, joten seuraavana ideana oli käyttää viivojen sijasta putkia, jotka voisivat muodostaa erilaisia putkistoja. Tämä vaikutti jo hyvältä idealta, joka johti ajatuksiin erilaisista verkostoista ja viemäreistä. Syntyneet ideat vaihtelivat vesiputkista sähköverkkoihin, kunnes idea valokaapeleista ja lasereista syntyi. Pelin idea oli valmis:

- 2D-Puzzle-peli, jossa pelaaja järjestelee erilaisia palikoita ja ohjaa niiden avulla lasersädettä paikasta toiseen. - Dr. Laser

Lasersäteiden lisääminen peliin avasi monia mielenkiintoisia mahdollisuuksia oikean ratkaisun kehittämiseksi. Lasersäteen ei tarvitse kulkea pelkästään putkissa vaan se voi tarvittaessa loikata ilman lävitse tai heijastua peilin avulla toiseen suuntaan. Pelin lopulliseksi tavoitteeksi syntyi siis lasersäteen ohjaaminen maaliin erilaisten komponenttien avulla.

3.4 Teeman suunnittelu

Pelit eivät itsessään vaadi teemaa, eikä kaikissa peleissä ole teemaa ollenkaan. Tällaisten pelien sanotaan olevan abstrakteja. Abstrakteihin peleihin teeman lisääminen tai vaihtaminen on yleensä helppoa ja onnistuu muuttamatta pelin mekaniikkoja. Teemoitetuilla peleillä tarkoitetaan puolestaan pelejä, joiden kaikki mekaniikat tukevat pelin teemaa. Pelin teema on tällöin hankalampi vaihtaa, mutta se ei ole vahvasti kytkettynä pelin mekaniikkaan. Tällaiseen peliin teeman lisääminen tai vaihtaminen jälkikäteen onnistuu kuitenkin vielä samankaltaisten teemojen kesken. On olemassa kuitenkin pelejä, jotka ovat saaneet alkunsa mekaniikkojen sijaan itse teemasta. Vahvasti teemoitetut pelit rakentuvat usein teemansa päälle, jolloin mekaniikat on räätälöity juuri kyseiseen teemaan sopivaksi. Tällöin pelin teeman vaihtaminen tai muuttaminen on hyvin hankalaa jälkikäteen ja vaatii usein pelimekaanisia muutoksia. [1; 2.]

Puzzlepelit ovat yleensä luonteeltaan abstrakteja eikä Sampo Lock ole tästä poikkeus. Sampo Lock perustuu ainoastaan tiettyihin mekaniikkoihin, jotka eivät olleet sidoksissa mihinkään tiettyyn teemaan. Tällöin teeman rakentaminen muuten valmiin pelin ympärille on huomattavasti helpompaa.

Teemat antavat peleille syvyyttä ja luovat tunnelmaa. Pelaajat pääsevät helpommin uppoutumaan pelin maailmaan, mikäli peli on hyvin teemoitettu. Teemat voivat myös parantaa pelin opittavuutta ja auttaa pelaajaa ymmärtämään erilaisia monimutkaisia pelimekaniikkoja. Pelaajat myös muistavat hyvin teemoitettuja asioita paremmin kuin täysin abstrakteja käsitteitä. Esimerkiksi pelaajan on helpompi muistaa ja ymmärtää, että ritari voittaa talonpojan kuin että kolmio voittaa ympyrän. [1.]

Suunnittelemani pelin abstraktista luonteesta johtuen peliin voidaan lisätä uusia teemoja helposti jälkikäteen. Teemoituksen kannalta ongelmallisin mekaniikka on pelin lasersäde. Peli istutettiin kuitenkin onnistuneesti jo olemassa olevan IP:n päälle nimeltä Aurora Nord. Aurora Nordin teema rakentuu modernisoidun Kalevalan ympärille, josta löytyi ennestään elokuvan käsikirjoitus ja paljon konseptitaidetta. [15.]

Ensimmäinen vaihe teeman luomiseen on aina tehdä tutkimustyötä ja käydä kaikki olemassa oleva materiaali läpi. Tässä kyseisessä tapauksessa se tarkoitti Kalevalaan paneutumista, elokuvan käsikirjoituksen lukemista ja konseptitaiteeseen tutustumista. Tutkimuksessa nousi esille yksi teemalle hyvin keskeinen esine eli Sampo. Sampo on seppä Ilmarisen takoma mystinen artefakti, joka soveltui pelin teemaksi erinomaisesti. Niinpä pelin nimeksi annettiin ”Sampo Lock”, laatikot muutettiin metallisiksi lukon osiksi, laserit maagisiksi säteiksi ja peilit kristalleiksi. Pelin uusi metallinen ja mystinen teema oli saanut alkunsa. Inspiraatiota uudelle ”look and feelille” haettiin myös reaalimaailman metallisista pulmapeleistä.

Kuva 3. Metallisia pulmapelejä, joita käytettiin uuden ”look and feelin” inspiraationa

Kuva 4. Google Playhin tehty kuva pelin uudesta teemasta

3.5 Pelimekaniikan suunnittelu

Peliin suunniteltiin useita erilaisia komponentteja ja mekaniikkoja, joista osaa ei koskaan toteutettu tai ne poistettiin lopullisesta versiosta. Pelimekaniikkojen suunnittelussa on tärkeää, että yksittäisten mekaniikkojen sijasta suunnitellaan kokonaisuuksia. Kaikkien mekaniikkojen tulee toimia hyvin yhdessä, jolloin saadaan aikaan tasapainoinen ja toimiva kokonaisuus. Kaikkea mieleen tulevaa ei kannata änkeä peliin väkisin. [1.]

Sampo Lockissa on 36 erilaista komponenttia, joista pelikenttä koostuu. Erilaisia mekaniikkoja on olemassa kuitenkin vain kuusi ja erilaiset ominaisuudet ovat vain näiden mekaniikkojen erilaisia yhdistelmiä. Käytetyt pelimekaniikat ovat:

- Liikuteltavuus, mahdollistaa komponentin liikuttamisen pelikentällä.
- Laserin generointi, synnyttää uuden lasersäteen tyhjästä.
- Lasermaali, pelin voittoehto täyttyy vasta kun jokaiseen lasermaaliin on ohjattu lasersäde.
- Laserin halkaiseminen, halkaisee olemassa olevan lasersäteen useaksi eri säteeksi.
- Laserin heijastaminen, vaihtaa laserin kulkusuuntaa.
- Laserin torjuminen, estää laserin kulkemisen johonkin tai kaikkiin suuntiin.

- Laserin tunneloiminen, siirtää laserin paikasta toiseen tunneloimalla.

Kuva 5. Sampo Lockissa käytetyt komponentit erilaisine ominaisuuksineen

Suunniteltuja ja poistettuja mekaniikkoja, jotka vaikuttivat aluksi hyviltä, mutta eivät toimineet lopullisessa kokonaisuudessa:

- Laserhäiritsin, pelin voittoehto täyttyy vasta, kun yhteenkään laserhäiritsimeen ei ole ohjattu lasersädettä. Käytännössä lasermaalin vastakohta.
- Portti, nostaa tai heikentää lasersäteen tehoa, jolloin voittoehto täyttyy vain, jos lasersäteellä on tarpeeksi tehoa.
- Prisma, hajottaa lasersäteen erivärisiksi lasersäteiksi, jolloin lasermaalit voisivat vaatia tietyn väristä laseria toimiakseen.

3.6 Käyttöliittymän suunnittelu

Mobiiliohjelmien käyttöliittymäsuunnittelu eroaa suuresti muiden alustojen käyttöliittymäsuunnittelusta, vaikka monet suunnittelun periaatteet pysyvätkin samoina. Käyttöliittymäsuunnittelun kannalta merkittävin ero mobiililaitteilla muihin alustoihin verrattuna on se, että lähes kaikki käyttäjän antamat syötteet tulevat kosketusnäytön kautta. Mobiililaitteiden näytöt ovat myös yleensä muita alustoja merkittävästi pienempiä, mikä jättää hyvin vähän tilaa käyttöliittymäkomponenteille.

Mobiililaitteille on ehtinyt jo kehittyä ja vakiintua useita omia käyttötapoja ja periaatteita, joita käyttäjät ovat oppineet tunnistamaan ja käyttämään. Tällaisia tunnettuja

käyttöliittymäratkaisuja ovat esimerkiksi listojen vieritys alaspäin sormea ylöspäin pyyhkäisemällä ja näkymän suurentaminen kahta sormea toisistaan eroon raahaamalla. Entuudestaan tunnettuja käyttöliittymäratkaisuja kannattaa suosia, koska ne parantavat pelin opittavuutta. [13.]

Pelien käyttöliittymäsuunnittelussa kannattaa myös kiinnittää erityistä huomiota pelien tulevaan lokalisointiin. Tekstit ovat hankalasti lokalisoitavissa, koska ne vaativat aina käännöstyötä ja erilaiset tekstien pituudet aiheuttavat usein ongelmia käyttöliittymän muotoilun kanssa. Pitkät tekstit vievät myös paljon tilaa pieneltä näytöltä, joten niitä kannattaa välttää. Tämän vuoksi symboleiden käyttö mobiilikäyttöliittymien suunnittelussa kannattaa maksimoida. Symbolit ovat selkeitä, nopeasti luettavia, vähän tilaa vieviä eivätkä vaadi lokalisointia.

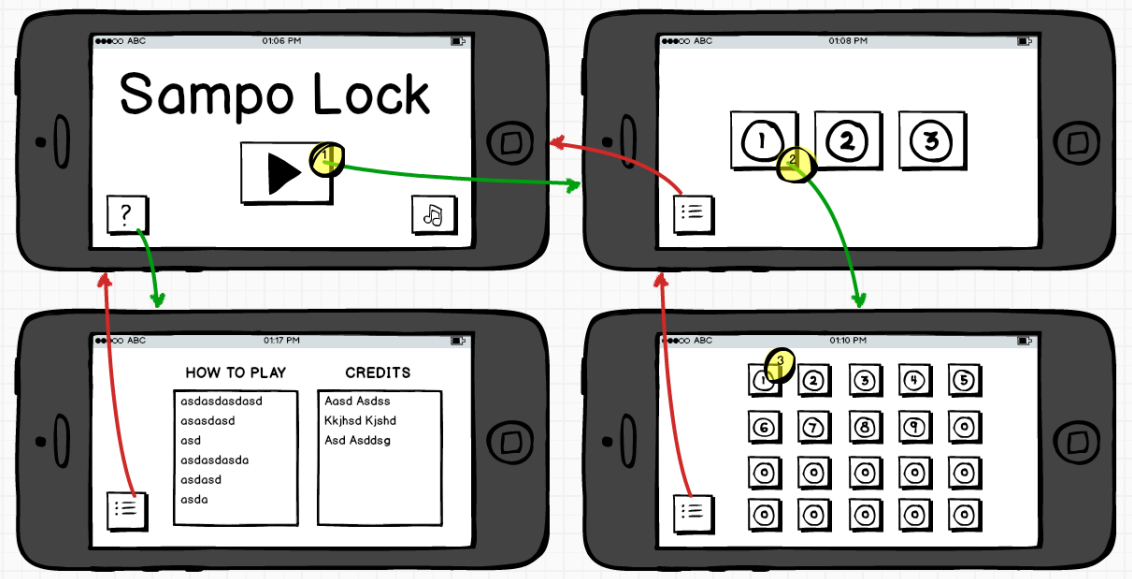
Valikoiden suunnittelu jätetään usein sivuun vähemmän tärkeänä asiana, mutta näin ei kuitenkaan saisi käydä. Pelikokemus alkaa heti pelin käynnistymisestä ja loppuu vasta, kun peli on kokonaan suljettu. Hyvät valikot ovat osa hyvää pelikokemusta. Valikoiden päätehtävänä on yleensä auttaa pelaajaa navigoimaan pelin asetusten ja pelimaailman välillä, mutta insinöörimäisestä tarkoituksestaan huolimatta valikoiden ei tarvitse olla ”insinöörimäisiä”. Insinöörimäisillä valikoilla tarkoitetaan yleensä tyylitöntä, vaikeasti navigoitavaa, liikaa informaatiota antavaa ja ironisesti loppukäyttäjän mielestä ”epäloogista” valikkoa. Tällaiseen valikkoon yleensä päädytään, mikäli valikkoa ei suunnitella kunnolla tai sen suunnittelu jätetään viime tinkaan. Valikon suunnittelu on tärkeä osa pelisuunnittelua ja valikkoa voi käyttää oikean pelitunnelman luomiseen jo ennen varsinaisen pelin alkamista. [1; 2.]

Valikkosuunnittelu kannattaa aloittaa ensin kartoittamalla, mitä pelaajan tulee voida tehdä valikossa. Näistä asioista voidaan luoda käyttötapauksia, jotka otetaan huomioon valikkoa suunniteltaessa. Käyttötapaukset Sampo Lockin valikoihin ovat:

- Pelaajan tulee voida selata pelattavia kenttiä.
- Pelaajan tulee nähdä, mitkä kentät hän on jo ratkaissut ja mitkä kentät ovat vielä ratkaisematta.
- Pelaajan tulee voida valita ja käynnistää haluamansa kenttä pelattavaksi.
- Pelaajan tulee voida kytkeä äänet päälle tai pois päältä.
- Pelaajan tulee voida löytää lisäohjeita pelin tavoitteista tarvittaessa.

- Pelaajan tulee voida löytää lisätietoa pelin tekijöistä.

Valikon toimivuus ja opittavuus kannattaa selvittää testaamalla jo varhaisessa vaiheessa prototyyppien avulla. Ensimmäiset prototyypit voidaan tehdä kokonaan paperista ja niitä voidaan testata suoraan jopa loppukäyttäjillä. Vaikka paperiproton tekeminen on hyvin nopeaa, niin voi sen interaktiivinen testaaminen olla hidasta ja vaivalloista. Tämän vuoksi paperiprotojen testaus jää yleensä vähäiseksi ja rajoittuu yleensä vain muutamaan testaajaan. Interaktiivisen prototyypin rakentaminen onnistuu kuitenkin suhteellisen helposti Androidin omilla työkaluilla tai vaikka MS PowerPointilla. Onneksi nykyään löytyy myös monia hyviä sovelluksia interaktiivisten prototyyppien rakentamiseen nopeasti ja vaivatta. Tällaisia sovelluksia ovat esimerkiksi NinjaMock, LucidChart ja Balsamiq Mockups. [11.]



Kuva 6. Balsamiq Mockupsilla tehty prototyyppi Sampo Lockin valikosta

Sampo Lockin valikoissa pyrittiin välttämään hankalasti käytettäviä listoja ja suosimaan suuria nappeja. Tunnettujen symboleiden käyttö pyrittiin myös maksimoimaan ja tekstien käyttö puolestaan minimoimaan. Pelin aloittaminen haluttiin myös tehdä mahdollisimman suoraviivaiseksi.

3.7 Kenttien suunnittelu

Kenttien suunnittelu on hyvin erilaista ja vaihtelee peleittäin suuresti, mutta niiden suunnittelu on aina työläs prosessi. Kaikissa peleissä ei kuitenkaan aina ole edes

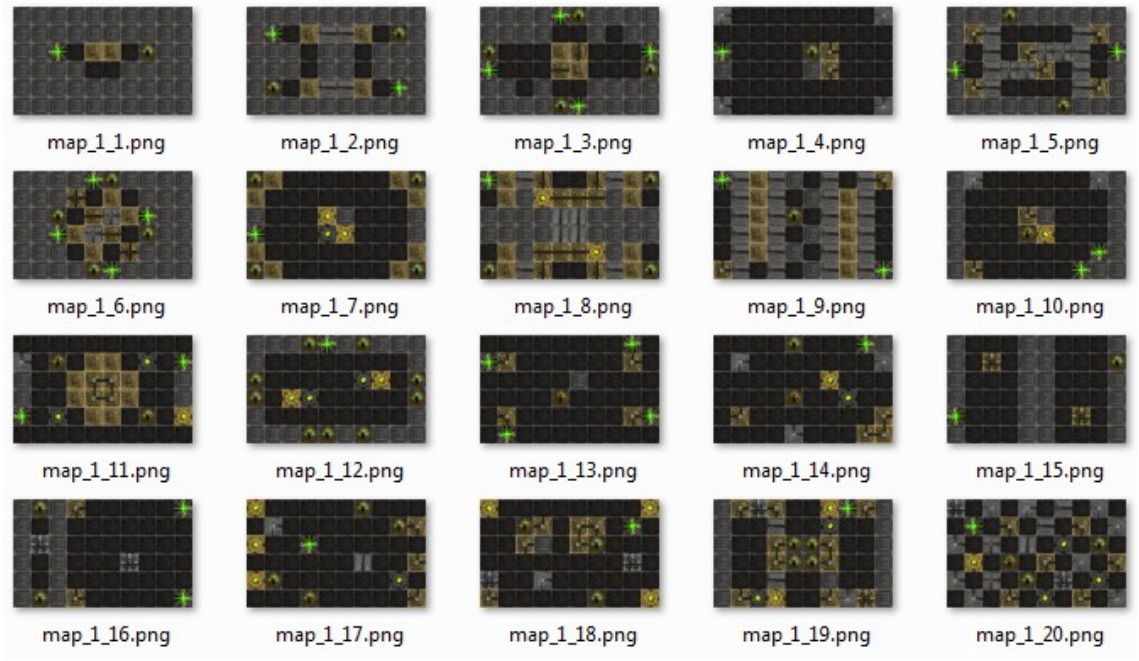
varsinaisia kenttiä tai ne voivat olla automaattisesti generoituja. Prototyyppien perusteella voitiin arvioida, että Sampo Lockkiin pitäisi suunnitella ainakin kuusikymmentä kenttää, jotta peliin saataisi tarpeeksi sisältöä. Voitaisiinko kentät generoida automaattisesti?

Automaattinen kenttien generointi olisi ollut teknisesti hyvin haastavaa rakentaa, koska kentät ovat suhteellisen monimutkaisia. Lisäksi olisi hyvin vaikeaa varmistaa, että automaattisesti generoidut kentät olisivat mielenkiintoisia, visuaalisia ja selkeitä eivätkä ne olisi liian haastavia tai helppoja. Kenttien suunnittelu täytyi siis tehdä käsin.

Hyvällä kenttäsuunnittelulla voidaan parantaa pelin opittavuutta merkittävästi ja pelaajalle voidaan opettaa uusia asioita, niin että pelaaja ei edes itse tiedosta asiaa. Pelaajan kannalta on myös paljon mukavampaa keksiä itse uusia asioita, esimerkiksi kuinka ovi avataan. Pelin ei tarvitse välkyttää pelaajalle tekstiä ja ääntä ”käännä oven kahvasta”, jos kenttäsuunnittelu on tehty hyvin. Riittää, että uusien asioiden oppiminen on tehty pelaajalle helpoksi. Tärkeää on välttää useiden uusien asioiden yhtäaikainen esittäminen ja antaa pelaajalle aikaa oppia yksi uusi asia kerrallaan. [2.]

Opittavuutta silmälläpitäen kentät jaettiin kolmeen vaikeus luokkaan, joista helpoin toimii eräänlaisena tutoriaalina. Pelin ensimmäiset kaksikymmentä kenttää suunniteltiin niin, että joka toinen kenttä esittelee pelaajalle uuden mekaniikan yksinkertaisessa ympäristössä, jonka jälkeen seuraavassa kentässä opittua uutta mekaniikkaa sovelletaan vaikeammassa ympäristössä. Näin pelaaja ehtii tutustua ja sisäistää kaikki pelin mekaniikat huomaamattaan ilman tyrkyttävää opastusta. Loppupään vaikeampien kenttien suunnittelu voitiin puolestaan aloittaa vaikean ratkaisun keksimisestä, koska opittavuudesta ja visuaalisuudesta ei enää tarvinnut huolehtia.

Ensimmäisten kenttien testaaminen oli helppoa ja niiden hiomiseen käytettiin enemmän aikaa kuin loppupään vaikeampien kenttien testaamiseen. Vaikeampien kenttien testaamisen ongelmaksi muodostui niiden huomattavasti hitaampi testaaminen ja hajanaiset testaustulokset. Osa testaajista piti joitain kenttiä vaikeina ja toiset taas helppoina, joten kenttien järjestäminen helpoista vaikeampiin oli hakalaa. Ensimmäisen version julkaisun jälkeen dataa alkoi kuitenkin kertyä enemmän ja ongelmakohtiin voitiin puuttua paremmin.



Kuva 7. Sampo Lockin ensimmäiset kaksikymmentä kenttää alkuasetelmissaan

4 Pelin toteutus

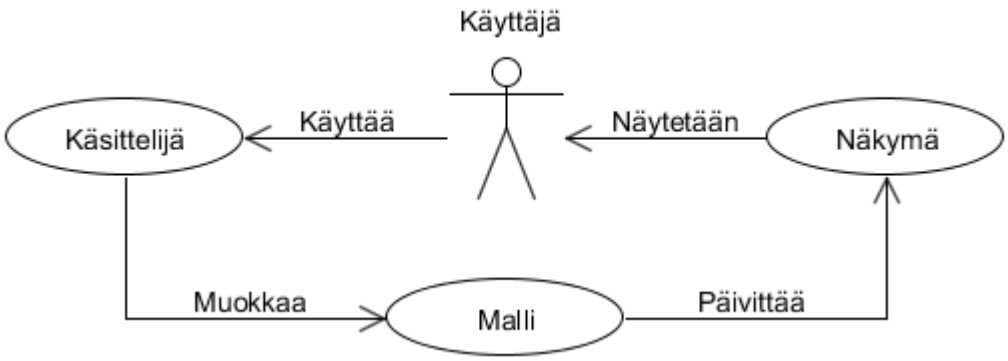
4.1 Pelin arkkitehtuuri

Tekninen toteutus pyrittiin pitämään mahdollisimman selkeänä ja yksinkertaisena, jotta peliä olisi helppo ylläpitää ja laajentaa. Riippuvuus Androidin ulkopuolisiin kirjastoihin haluttiin myös minimoida samoista syistä. Turhan suuret ja monimutkaiset kolmannen osapuolen kirjastot usein päätyvät vain monimutkaistamaan yksinkertaisia asioita, jos ne on otettu käyttöön turhaan. Ulkopuolisten kirjastojen käyttäminen voi nopeuttaa koodin kirjoittamista, mutta ne myös hämärtävät kehittäjän kokonaisvaltaista ymmärrystä ohjelman rakenteesta. Usein uuden kirjaston ja sen rajapintojen opiskeluun voidaan käyttää enemmän aikaa kuin itse ongelman ratkaisemiseen. Mitä paremmin ja syvemmin kehittäjät ymmärtävät projektinsa koodia, niin sitä parempaa koodia he voivat kirjoittaa. Koodin optimointi ja virheiden korjaaminen on tällöin myös huomattavasti helpompaa. Kolmannen osapuolen kirjastoja ei siis kannata ottaa käyttöön kevyin perustein. Ulkopuolisia kirjastoja tulee käyttää ainoastaan, mikäli niistä on todellista hyötyä projektille.

Sampo Lock on toiminnaltaan hyvin yksinkertainen eikä sen toteuttamiseen katsottu tarpeelliseksi käyttää mitään ulkopuolisia kirjastoja. Arkkitehtuurissa haluttiin kuitenkin

erottaa näkymän toteutus muusta koodista, koska sen toteuttamiseen oli useampia eri vaihtoehtoja. Vaihtoehdoista suorituskykyisin olisi ollut käyttää Androidin tukemaa OpenGL ES-piirtokirjastoa. OpenGL on kuitenkin 3D-piirtokirjasto, jolloin se on turhan monimutkainen yksinkertaisten 2D-grafiikoiden piirtämiseen. Yksinkertaisempi ja helpompi ratkaisu oli käyttää Androidin tarjoamaa 2D-piirtorajapintaa, jolla pelinäkymä voitiin piirtää suoraan näkymän kanvakselle. Androidin 2D API ei ole kovin tehokas tai monipuolinen, mutta se osoittautui kuitenkin täysin riittäväksi tälle projektille. [18.]

Edellämainituista syistä pelin arkkitehtuurissa pyrittiin noudattamaan MVC-ohjelmistoarkkitehtyyrityyliä, joka kuuluu tunnettuihin hyviin suunnittelumalleihin. MVC tulee sanoista Model View Controller ja sen tarkoituksena on käyttöliittymän erottaminen muista sovellusalustan tiedoista. Suunnittelumallin etuna on myös se, että pelin mallin voi suunnitella, ohjelmoida ja testata riippumatta järjestelmän muista osista. Tiukan MVC-arkkitehtuurin noudattaminen ei ole kaikkein tehokkain ratkaisu, eikä malli itsessään ohjaa kehittäjää koodin optimointiin. Koodin selkeyttä ei kuitenkaan kannata uhrata suoritustehokkuuden takia enää nykypäivänä.

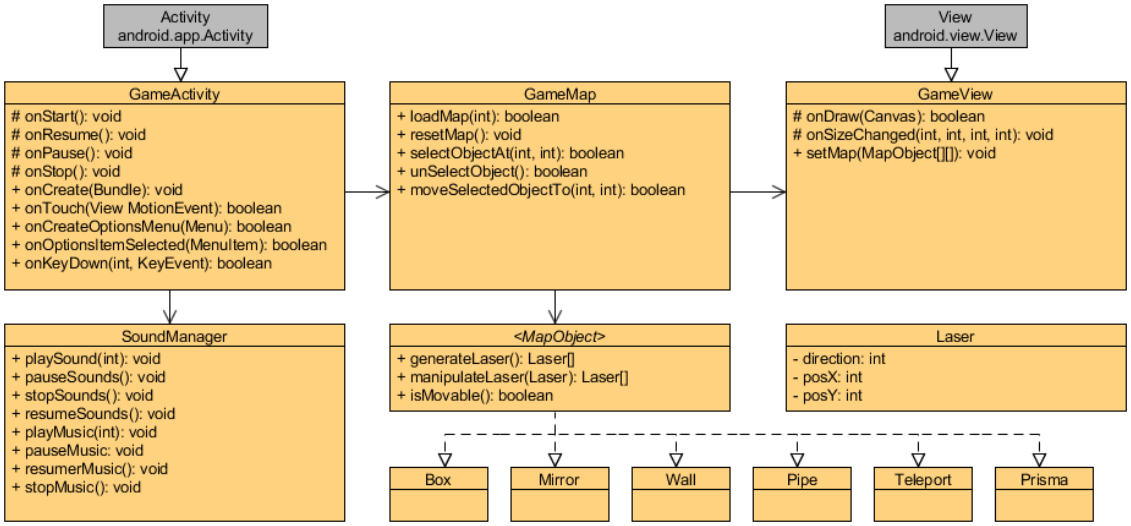


Kuva 8. MVC-arkkitehtuurityylin rakenne

Sampo Lockin keskeisimmät kolme luokkaa ovat GameActivity, GameMap ja GameView. GameActivity on pelin käsittelijä, joka periytyy Androidin Activitysta ja hallitsee käyttöjärjestelmältä tulevia tapahtumia, kuten pelin käynnistäminen ja sulkeminen. GameActivity käsittelee myös käyttäjältä tulevia syötteitä toteuttamalla OnTouchListener rajapinnan ja muokkaa pelin mallia tarvittaessa.

Pelin malli koostuu useammasta eri luokasta jotka on kuitenkin kapseloitu GameMap luokan alle. GameMap on vastuussa pelin keskeisestä logiikasta ja tärkeimmistä mekaniikoista, kuten palikoiden ja lasersäteiden liikuttaminen. Pelitilan säilyttäminen ja ylläpitäminen kuuluu myös mallin tehtäviin. Pelitilan muuttuessa GameMap ilmoittaa

asiasta pelin näkymälle, eli GameViewille. GameView huolehtii pelitilan visuaalisesta esittämisestä pelaajalle. GameView periytyy Androidin Viewistä, jolloin se on helppo vaihtaa tarvittaessa toiseen Viewiin.



Kuva 9. Sampo Lock pelin keskeisimmät luokat ja metodit

4.2 Käyttöliittymän toteutus

Androidille on kaksi perustapaa toteuttaa käyttöliittymä. Mikäli haetaan maksimaalista suoritustehokkuutta, niin UI-elementit kannattaa piirtää samaan OpenGL-piirtopintaan kuin itse pelinäkymä. Tämä vaatii kuitenkin usein oman tilakoneen ja UI-elementtien rakentamista tai ulkopuolisten kirjastojen käyttämistä. Oman UI-ratkaisun kehittäminen ja käyttäminen voi myös merkittävästi helpottaa käännöstyötä, mikäli peli aiotaan myöhemmin kääntää muille alustoille.

Toinen tapa käyttöliittymän rakentamiseen on käyttää Androidin tarjoamia valmiita UI-elementtejä ja Layoutteja. Sampo Lock ei ole reaaliaikainen peli eikä se vaadi suorituskyvyn optimoimista. Pelinäkymän piirtämiseen ei käytetä OpenGL:ää, joten sitä ei kannata ottaa käyttöön vain valikkojen takia. Sampo Lockin käyttöliittymä voitiin toteuttaa kokonaan Androidin omalla käyttöliittymäsysteemillä.

Layoutit eli sommittelut määrittelevät käyttöliittymän visuaalisen rakenteen. Layoutteja voidaan luoda ajanaikaisesti tai niitä voidaan määritellä erillisillä XML-tiedostoilla. Layouttien määrittely XML:ssä auttaa erottamaan käyttöliittymän tyylin sitä hallitsevasta koodista. Android tukee myös automaattista tyylin valintaa, jossa samalle näkymälle voi

olla useita eri tyylejä riippuen laitteen näytön koosta ja muodosta. Tämä on erityisen hyödyllistä mobiilimaailmassa, koska näyttöjen koot ja muodot vaihtelevat suuresti. Sama käyttöliittymä ei usein toimi pienessä puhelimessa kuin suuressa tabletissa. [19.]

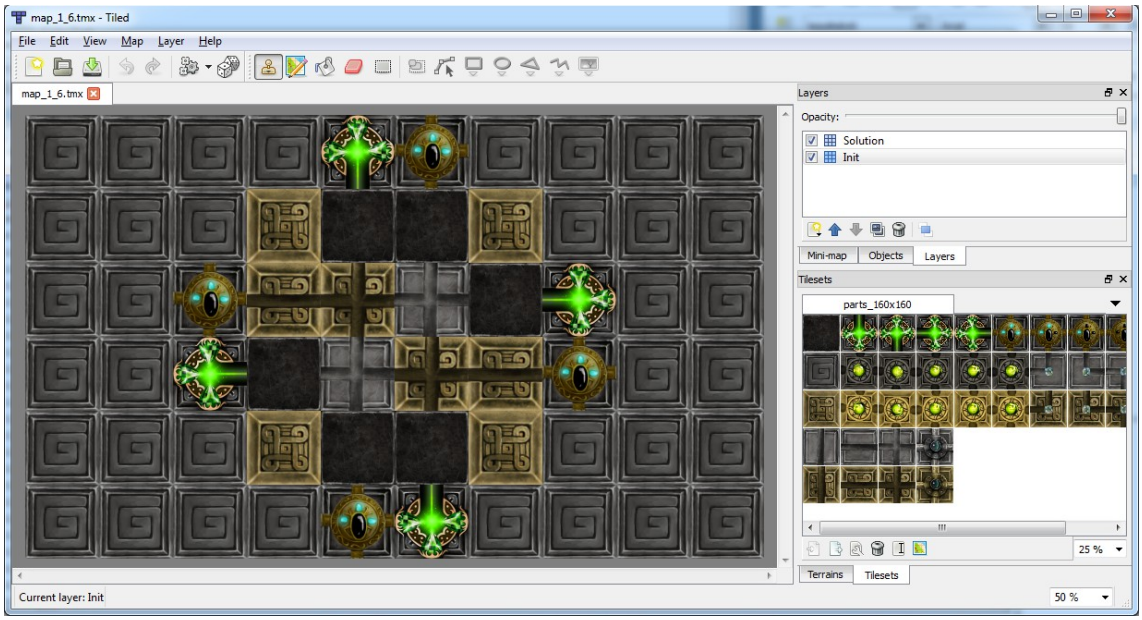
4.3 Kenttäeditori

Sisällön tuottaminen on usein yksi työläimpiä ja aikaavievimpiä vaiheita pelin kehityksessä. Suuremmissa peleissä pelaaja voi ohittaa minuuteissa sellaisen sisällön, jonka luomiseen oli käytetty satoja tunteja. Siksi on tärkeää ottaa sisällön luominen huomioon jo pelin suunnittelun yhteydessä ja pohtia miten luomisprosessia voitaisiin nopeuttaa ja helpottaa. Kenttäeditorien ja muiden vastaavien pelikehityksen sisäisten työkalujen rakentamiseen voi kulua paljon aikaa, mutta ne maksavat usein itsensä takaisin varsin nopeasti pitkäkestoisemmissa projekteissa. [2.]

Yksi merkittävä syy puzzlepelin valitsemiseen olikin juuri sisällön luomisen tehokkuus eli sisällön tekemiseen kulutettu aika on hyvässä suhteessa sen tuottamaan peliaikaan. Kenttätiedostojen luonti Sampo Lockiin on suhteellisen yksinkertaista, ja kenttätiedostot olisi voitu kirjoittaa jopa käsin. Kenttiä oli kuitenkin suunniteltu yli kuusikymmentä ja niiden käsinkirjoittaminen olisi ollut hidasta sekä virhealtista. En kuitenkaan halunnut käyttää paljon aikaa ”ei niin välttämättömän” työkalun luomiseen, joten lähdin etsimään olemassa olevia valmiita ratkaisuja.

Erilaisia kenttäeditoreita 2D-pelejä varten on tehty varmasti tuhansia, mutta suurien ja kaiken kattavien pelikehitysympäristöjen kuten Unityn myötä niiden päivitys on heikentynyt. Vielä on kuitenkin olemassa muutama ajantasalla oleva vaihtoehto tilepohjaisten kenttien rakentamiseen kuten: Tile Studio, tIDE ja Tiled Map Editor. Tile Studio tarjoaa kenttäeditorin lisäksi erilaisia 2D-piirtotyökaluja ja tIDE sisältää kokonaisen pienen pelikehitysympäristön. Tiled Map Editor on edelleen kehityksen alla, ja se on tarjolla olevista vaihtoehdoista parhaiten ylläpidetty. Tiled on kevyt ja yksinkertainen editori, joka tarjoaa ainoastaan kenttäeditorin kannalta oleellisimmat ominaisuudet kuten:

- yksinkertaisen käyttöliittymän kentän editoimiseen ja visualisoimiseen,
- tuen usempi kerroksiselle kenttätiedostoille ”Layers”,
- kenttätiedoston tallenneksen XML, JSON ja LUA formaatteihin. [8; 9; 10.]



Kuva 10. Tiled Map Editorin päänäkymä

Jokainen kenttä koostuu kahdesta tasosta: aloitustilanteesta ja esimerkkiratkaisusta. Aloitustilanne on tilanne, josta uusi pelaaja aloittaa kentän ratkaisemisen. Esimerkkiratkaisu on yksi mahdollinen ratkaisu kentälle, jota käytetään kentän validoimiseen ja vihjeiden antamiseen pelaajalle.

Tiled Map Editori perustuu tileihin. Tilet ovat erinäköisiä, mutta samankokoisia pieniä kuvia, joita yhdistämällä toisiinsa kenttätiedosto luodaan. Tilet eli laatat pilkotaan yleensä yhdestä suuremmasta kuvasta, jota nimitetään atlakseksi. Kentän luomiseen Tiled Map Editor tarvitsee tiedon kentän koosta tileinä sekä yksittäisen tilen koon pikseleinä. Sampo Lockissa kentät ovat kymmenen laattaa leveitä ja kuusi korkeita. Yksittäisen laatan alkuperäinen koko on 160x160 pikseliä, mutta pelissä laattojen koko sovitetaan aina kyseisen näytön mukaan. [8.]

Tiled Map Editor voi tallentaa kentät useaan eri formaattiin. Päädyin kuitenkin käyttämään XML-formaattia, koska Androidilla on erinomainen luontainen tuki sen käsittelyyn. Androidin tarjoama XmlPullParser on yksinkertainen, mutta tehokas rajapinta XML-tiedostojen käsittelyyn. Android on kehittänyt omalaatuisen, mutta käytännöllisen systeemin käyttöliittymien skriptaamiseen XML:n avulla erottaakseen tyylin ja lähdekoodin toisistaan. Android perustuu pitkälti tämän XML-rajapinnan ajonaikaiseen suorittamiseen monissa tehtävissä. Ajonaikainen XML:n parsiminen ja lukeminen on kuitenkin hirvittävän hidasta, johon Android onkin kehittänyt oivan ratkaisun. Android esikäsittelee ja tallentaa XML-tiedostot binäärinä ohjelman käännön

yhdeydessä. Tällöin XmlPullParserin kautta näiden esikäsiteltyjen XML-tiedostojen käyttäminen on yli kymmenen kertaa nopeampaa kuin normaalisti. [12]

```
<?xml version="1.0" encoding="UTF-8"?>
<map version="1.0" orientation="orthogonal" width="10" height="6" tilewidth="160" tileheight="160">
 <tileset firstgid="1" name="parts" tilewidth="160" tileheight="160">
  <image source="xml\parts.png" width="1600" height="800"/>
 </tileset>
 <layer name="Init" width="10" height="6">
  <data>
   <tile gid="11"/>
   <tile gid="3"/>
   ...
  </data>
 </layer>
 <layer name="Solution" width="10" height="6">
  <data>
   <tile gid="11"/>
   <tile gid="3"/>
   ...
  </data>
 </layer>
</map>
```

Kuva 11. Esimerkki XML-kenttätiedostosta.

Androidin XML-systeemin heikkoutena on kuitenkin se, että binääriksi käännettyjen XML-tiedostojen muokkaaminen ajonaikana ei onnistu. Käännettyjä XML-tiedostoja ei myöskään voida enää lukea tekstinä, vaan niiden lukeminen joudutaan tekemään aina XmlPullParserian kautta. [12.]

Androidin tarjoama XmlPullParser soveltuu kuitenkin Sampo Lockin tarpeisiin mainiosti, jonka lisäksi se nopeuttaa kenttien lataamista ja vaikeuttaa ratkaisujen urkkimista apk-pakkauksen sisältä.

4.4 Sallittujen siirtojen validointi

Peli perustuu neliöiden muotoisten palikoiden liikuttamiseen ahtaassa ympäristössä. Pelin tärkein yksittäinen mekaniikka onkin näiden palikoiden raahaaminen liu'uttamalla paikasta toiseen. Ilman tätä ominaisuutta peliä ei voisi pelata, joten ominaisuus piti toteuttaa ensimmäisenä ja tarjosi ensimmäisen haasteen projektille.

Palikoiden liikuttaminen tulee noudattaa seuraavia sääntöjä:

- Palikan voi liikuttaa ainoastaan tyhjään ruutuun, jossa ei ole toista palikkaa.
- Palikka ei saa kulkea toisten palikoin yli eikä lävitse.

Kuva 12. Vihreät nuolet osoittavat sallittuja- ja punaiset kiellettyjä siirtoja

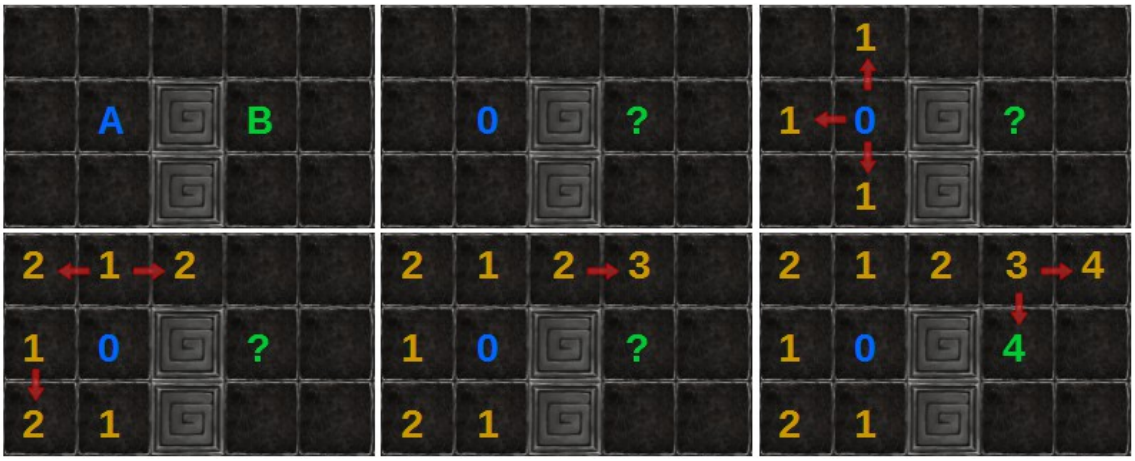
Ensimmäisen säännön tarkistaminen on helppoa. Voimme vain tarkistaa, onko ruutu, johon palikkaa halutaan siirtää, tyhjä vai ei. Tarkistuksen tekninen toteuttaminen koodissa on myös yksinkertaista, koska kulloinenkin pelitila on aina tiedossa. Ilman ajan tasalla olevaa tietoa pelin tilasta ei pelin näkymää voitaisi edes päivittää.

Toista sääntöä ei aluksi meinattu toteuttaa, koska se havaittiin turhaksi, mikäli palikkaa sallitaan liikuttaa vain yksi ruutu kerrallaan. Liikuttamisen rajoittaminen toimi hyvin ahtaissa kentissä, joissa ei ollut paljon tilaa palikoiden liikuttamiseen. Liikerajoitus osoittautui kuitenkin häiritseväksi ominaisuudeksi avaremmissa kentissä, joissa palikoilla oli paljon tilaa liikkua. Liikuttamisesta piti saada sujuvampaa, joten toisen säännön tarkistusta lähdettiin toteuttamaan.

Toisen säännön tarkastaminen on kuitenkin hieman hankalampaa, koska tarkastusta ei voida tehdä vain lähtö- ja kohderuudun perusteella. Toinen sääntö vaatii, että lähtöruudusta tulee päästä päämääränä olevaan ruutuun ilman, että kuljetaan yhdenkään toisen palikan lävitse. Kyseessä on siis mahdollisen reitin löytäminen kahden pisteen välillä. Tämä on yleinen ja hyvin tunnettu ongelma erilaisissa tietorakenteissa, joihin löytyy monia erilaisia ratkaisuita.

Mahdollisen reitin löytämiseen sovellettiin Sven Koenigin ja Maxim Likhachevin kehittämää D* Lite , eli ”D start lite”-algoritmia. D* Lite on yksinkertaistettu versio tunnetummasta D*-algoritmista, joka pohjautuu A*-algoritmiin. Kaikki mainitut algoritmit ovat pohjimmiltaan heuristisia ja inkrementaalisia hakualgoritmeja, jotka kaikki on alkujaan kehitetty ratkaisemaan autonomisten robottien kulkureittejä tuntemattomassa

maastossa. Vaikeista nimistään huolimatta D* Lite on hyvin yksinkertainen ja tehokas pieni algoritmi mahdollisten reittien etsimiseen. [16; 17.]



Kuva 13. Sallittun reitin etsimisessä käytetyn algoritmin toimintaperiaate

Sallittun reitin etsimiseen käytetty algoritmi toimii seurravalla tavalla:

- Asetetaan aloitusruutuun arvo 0 ja tallennetaan se muuttujan A arvoksi. Tämä tehdään vain kerran algoritmin alustuksen yhteydessä.
- Valitaan ruudukosta kaikki ruudut, joiden arvo on A.
- Kasvatetaan muuttujan A arvoa yhdellä ja asetetaan sen uusi arvo valittujen ruutujen viereisiin ruutuihin vain ja ainoastaan jos ne ovat tyhjiä.
- Jatketaan iteroimista, kunnes kohderuutuun on asetettu jokin arvo tai yhtäkään uutta arvoa ei lisätty ruudukkoon edellisessä iteraatiossa.
- Sallittu reitti on löytynyt mikäli iteroinnin loputtua kohderuudusta löytyy arvo. Muussa tapauksessa siirto on laiton.

4.5 Vihje toiminnon toteutus

Pelitestauksessa havaittiin jo kehityksen alkuvaiheilla, että peli tulee tarvitsemaan jonkinlaisen vihjetoiminnon. Puzzlepeleissä suurimpia turhautumisia aiheuttaa pelaajan juuttuminen johonkin tiettyyn heille hankalaan kohtaan. Tällöin pelaajan edistyminen voi tyrehtyä kokonaan, jolloin pelaaja menettää kiinnostuksensa hyvin nopeasti. Ihmiset tarrautuvat usein johonkin tiettyyn ajatusmalliin hyvin tiukasti, jonka jälkeen heidän on vaikea päästä siitä enää eroon. Mikäli ajatusmalli on alkujaan väärä, ei pelaaja tule löytämään oikeaa ratkaisua millään. Pelaaja pitäisi saada lähestymään ongelmaa uudelta kantilta. Vihjeet ovat olemassa juuri tätä tarkoitusta varten. Pienen vihjeen

antaminen voi johdattaa pelaajan oikealle tielle auttamalla pelaajaa näkemään ongelman uusin silmin. Vihje ei kuitenkaan saa olla liian paljastava, jotta peli ei menetä haasteellisuuttaan pelaajan silmissä. Mikäli pelistä tulee liian helppo tai pelaajaa on autettu liikaa, niin ei pelaaja koe enää onnistumisen tunnetta ratkaisun löydyttyä. [2.]

Sampo Lock peliin päätettiin kehittää ”Hint”-, eli vihjetoiminto, jonka tarkoitus on auttaa pelaajaa antamalla pelaajalle pieniä vihjeitä mahdollisesta ratkaisusta. Vihjeet eivät saisi olla liian paljastavia, mutta eivät myöskään hyödyttömiä. Toiminto haluttiin paljastavan pelaajalle aina vain yhden komponentin oikean paikan kerrallaan, jonka lisäksi toiminnon käytölle piti asettaa odotusaika, joka estäisi pelaajaa käyttämästä toimintoa liian useasti peräkkäin lyhyellä aikavälillä.

Kuinka tällaista ominaisuutta voitaisiin lähteä toteuttamaan? Yksi vaihtoehto olisi kehittää tekoäly ja joukko algoritmeja, jotka yhdessä laskisivat parhaan mahdollisen siirron nykyisen pelitilanteen pohjalta. Peli on kuitenkin huomattavan monimutkainen, ja suurimmaksi ongelmaksi muodostuvat toisistaan poikkeavat voittoehdot: jokainen kenttä on erilainen ja oikeita ratkaisuja voi olla useita. Algoritmien keksiminen ja kehittäminen on hankalaa ja aikaa vievää hommaa, jopa huomattavasti yksinkertaisempiin peleihin. Tällaisen tekoälyn kehittäminen ei olisi kannattavaa, koska se vaatisi liikaa työtä ja saattaisi vaatia lopulta liikaa laskentatehoa.

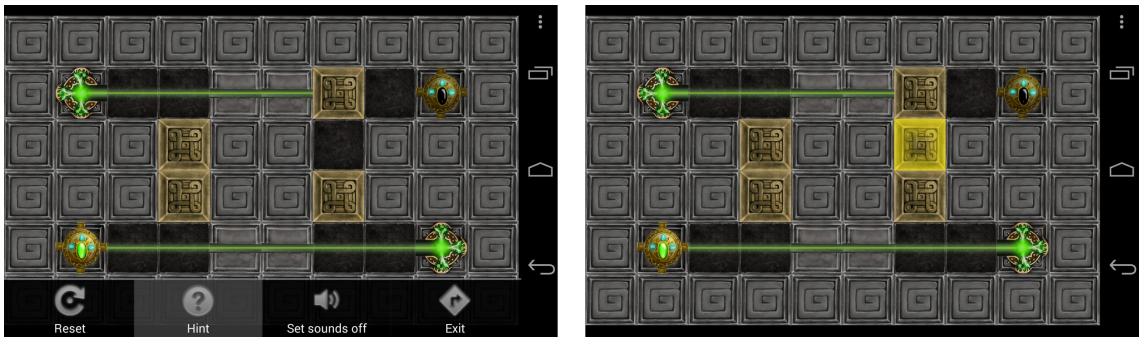
Seuraavana mieleen tullut vaihtoehto oli käyttää ”brute forse”-strategiaa ratkaisun löytämiseen. Strategiana olisi käydä kaikki kentän mahdolliset asetelmat läpi järjestyksessä ja lopettaa ensimmäisen ratkaisun löydyttyä. Koska kentät ovat kohtuullisen pieniä, niin strategia vaikuttaa mahdolliselta ratkaisulta. Mikäli mahdollisia asetelmia ei olisi aivan valtavasti, niin ensimmäisen ratkaisun löytäminen olisi suhteellisen nopeaa ja suoraviivaista. Liikuteltavia palikoita voi kuitenkin olla yhdessä kentässä peräti kuusikymmentä, joista eilaisia palikoita voi olla viisitoista. Tällöin mahdollisia yhdistelmiä on pahimmassa tapauksessa 15^60, eli tähtitieteellisiä määriä. Edes yhden ratkaisun etsiminen tällä strategialla saattaisi kestää todella kauan, jolloin etsimistä ei voida tehdä ajon aikana. Ratkaisut täytyy siis olla esilaskettuja. Tässä pääsemmekin kolmanteen ja lopulliseen ratkaisuun.

Kenttäsuunnittelun yhtenä osana on varmistua, että kentällä on olemassa ainakin yksi mahdollinen ratkaisu. Koska tämä ratkaisu on tiedossa, niin voimme tallentaa ratkaisun suoraan kenttätiedostaan ja käyttää sitä sellaisenaan vihjeen antamiseen pelaajalle.

Mitään esilaskentaa ei tällöin tarvita. Lisäksi kenttäsuunnitteljan kehittämä ratkaisu on todennäköisesti helpommin lähestyttävä ja loogisempi kuin mitä automaattisesti generoitu ratkaisu saattaisi olla.

Vihjetoiminnon lopullinen toteutus Sampo Lockissa toimii seuraavalla tavalla:

- Vertaa nykyistä pelitilaa ja esimerkkiratkaisua toisiinsa.
- Valitse ensimmäinen väärässä paikassa oleva liikuteltava objekti.
- Etsi ensimmäinen oikea paikka valitulle objektille, jossa ei vielä ole valitun kaltaista objektia.
- Tallenna valitun objektin nykyinen sijainti ja sen oikea paikka muistiin vihjeen antamista varten.



Kuva 14. Vihjetoiminnon käyttö

Mikäli käyttäjä painaa pelivalikosta löytyvää ”Hint”-painiketta, niin esitetään hänelle vihje oikeasta ratkaisusta visuaalisesti (kuva 13). Tällöin väärässä paikassa oleva komponentti piirretään keltaisena oikeaan paikkaan, mutta mitään komponentteja ei kuitenkaan liikuteta valmiiksi. Vihje katoaa heti, kun pelaaja tekee seuraavan siirtonsa.

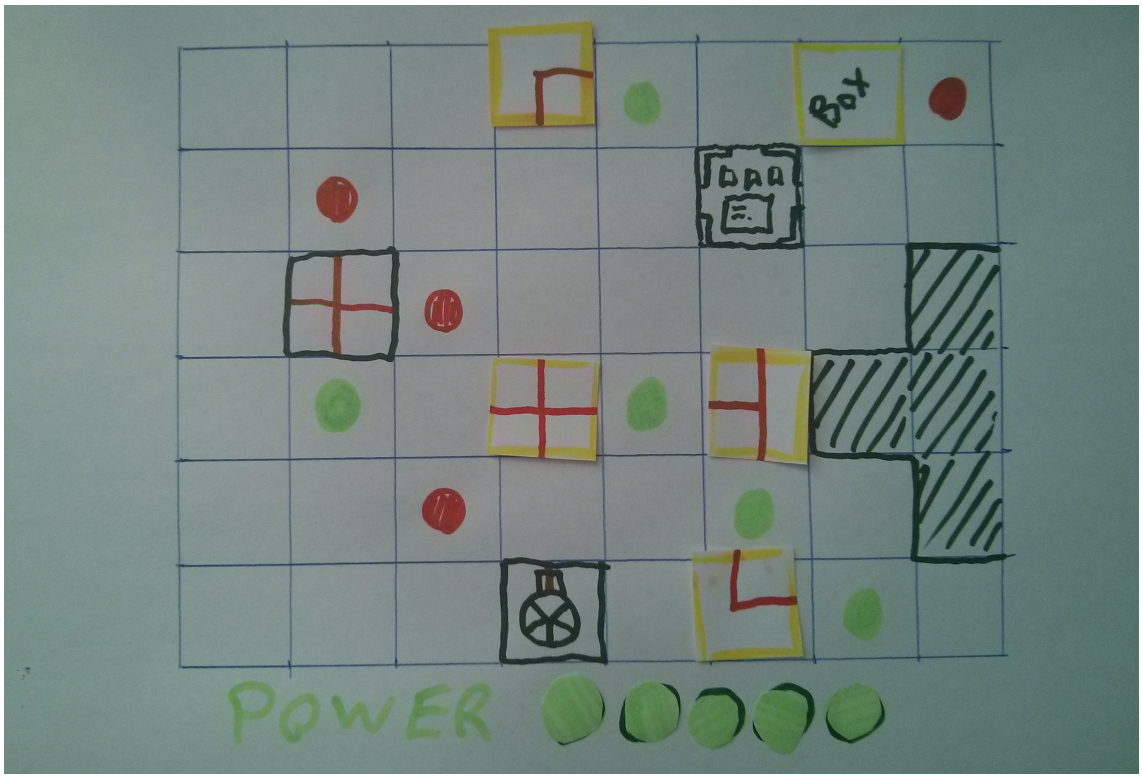
5 Pelin testaaminen ja iterointi

Testaaminen on hyvin tärkeässä osassa kaikissa ohjelmistoprojekteissa jo pelkästään ohjelman toimivuuden varmistamiseksi. Pelitestaaminen on kuitenkin erilaista testaamista, jossa tarkoitus on testata muun muassa pelin opittavuutta ja hauskuutta. Muista ohjelmistoprojekteista poiketen peliprojekteissa ei koskaan voida olla varmoja siitä, tuleeko asiakas olemaan tyytyväinen lopulliseen tuotteeseen, koska varsinaiset asiakkaat ovat usein loppukäyttäjiä eli pelaajia. Hyötyohjelmien asiakkailla voi olla jotain tietoa siitä, mitä he todella haluavat, mutta pelaajat eivät tiedä, mitä he todella

haluavat ennen kuin se on annettu heille. Pelaajat tahtovat ainoastaan hyviä pelejä. Pelitestausta ja jatkuvaa suunnittelua tehdäänkin koko pelin kehityksen ajan ensimmäisistä prototyypeistä viimeiseen versioon asti. Pelikehitykselle tyypillinen iterointiprosessi ja lopullinen yksityiskohtien ”viilaaminen” vie yleensä moninkertaisesti enemmän aikaa kuin varsinaisen pelin toteutus. Moni peliprojekti on epäonnistunut, vain, koska projektin alkumetreillä ei osattu varata tarpeeksi aikaa pelin viimeistelyä varten. Totuus on kuitenkin se, että lähes mistä vain lähtökohdista voidaan kehittää hyvä ja toimiva peli, kunhan peliä testataan, viilataan ja iteroidaan tarpeeksi. Pelien julkaisun kanssa ei tulisi koskaan kiirehtiä, koska suurin osa pelin tuotosta ja maineesta tehdään kahden ensimmäisen viikon sisällä julkaisusta. [1.]

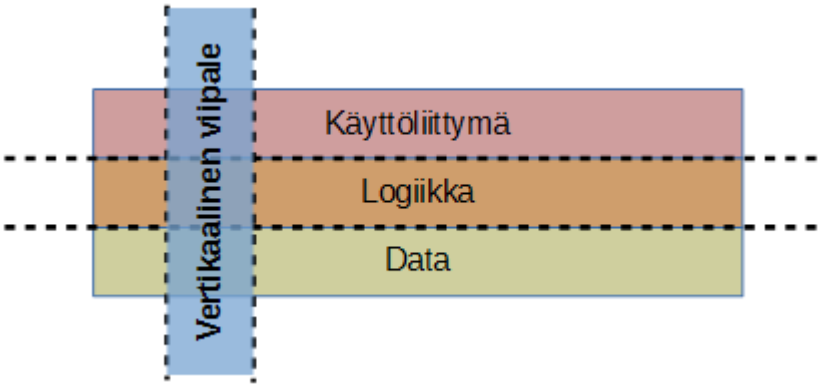
Jotta pelitestaaminen saadaan projektiin mukaan mahdollisimman varhaisessa vaiheessa, niin pelistä on tärkeää saada ensimmäinen prototyyppi nopeasti valmiiksi. Ensimmäisen prototyypin pohjalta on hyvä jatkaa pelin kehittämistä lisäämällä siihen uusia ominaisuuksia ja toiminnallisuutta pienissä osissa. Tämä on tyypillistä ketterissä kehitysmalleissa kuten Agility ja Scrum. Ketterät kehitysmallit soveltuvatkin erinomaisesti pelien kehitykseen.

Mikäli pelin pelimekaniikat ovat riittävän yksinkertaisia, niin voidaan peliä testata jo konseptitasolla ennen yhdenkään koodirivin kirjoittamista. Näin tehtiin myös Sampo Lockin kanssa, jolloin peliä päästiin testaamaan jo varhaisessa vaiheessa paperiprotojen avulla. Paperiprotoamista käytetään edelleen paljon erityisesti käyttöliittymäsuunnittelussa, mutta tekniikkaa kannattaa hyödyntää myös muualla mikäli mahdollista.



Kuva 15. Paperiproto suunnitellusta puzzlepelistä

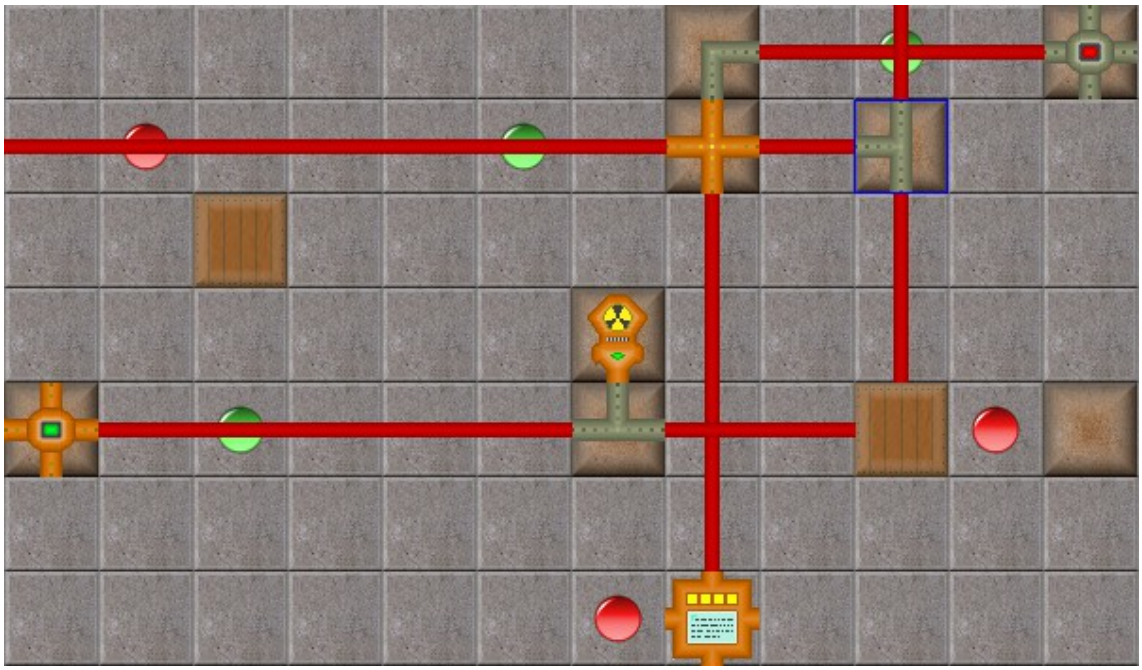
Paperiprotoilun jälkeen pelin varsinainen kehitys voidaan aloittaa kahdella eri tavalla. Pelistä kannattaa aluksi tehdä vain pieni vertikaalinen tai horisontaalinen viipale, jonka jälkeen kehitystä jatketaan aina siivu kerrallaan. Tällöin pelitestausta voidaan tehdä jatkuvasti ja tarvittaviin muutoksiin voidaan reagoida nopeasti ja mahdollisimman aikaisessa vaiheessa. [14.]



Kuva 16. Vertikaalinen viipaleesta ohjelmistosta

Ensimmäinen vertikaalinen viipale puzzlepelistä väliaikaisilla grafiikoilla oli jo täysin pelattava peli, mutta vain yhdellä kentällä. Peli oli kuitenkin kaukana valmiista, mutta ensimmäisen viipaleen toteuttaminen oli nopeaa ja iterointi voitiin aloittaa.

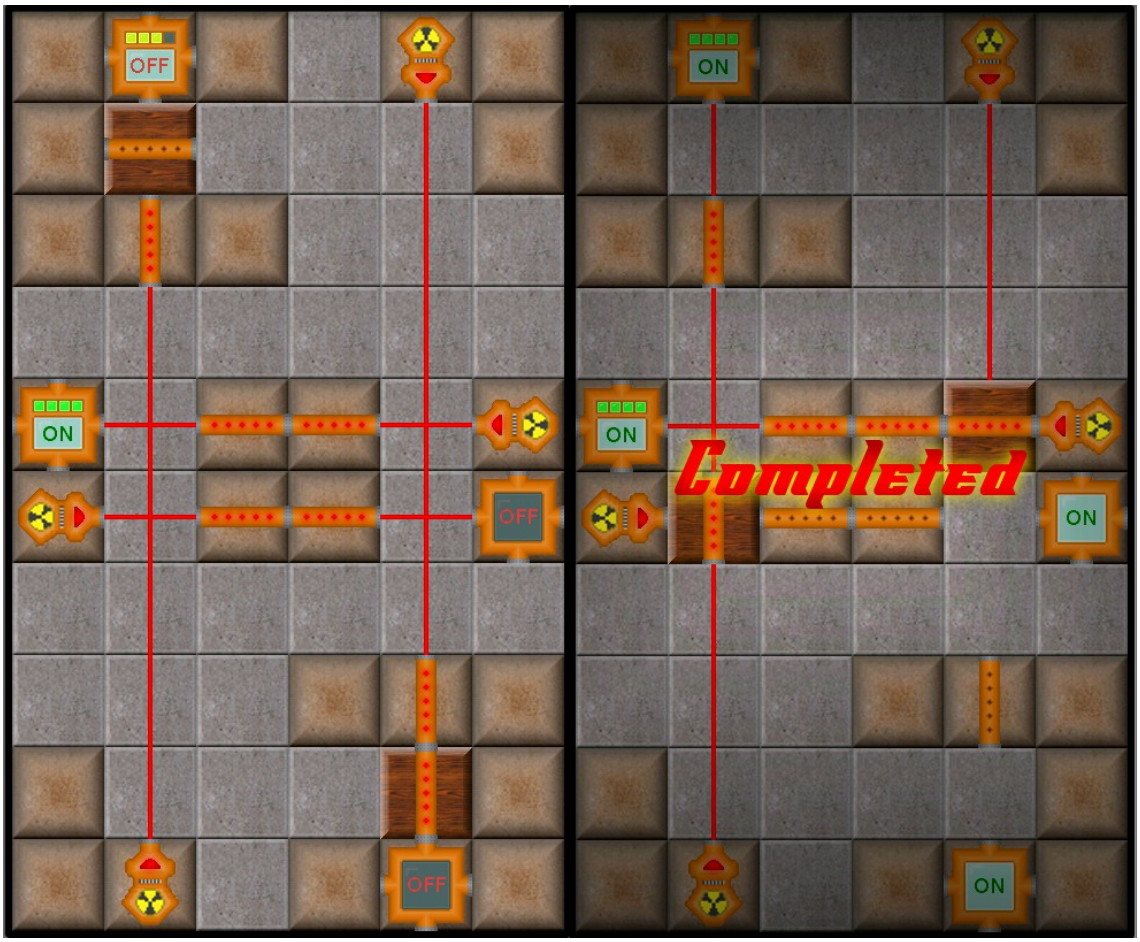
Ensimmäisen version pelitestauksessa havaittiin, että pelaajien oli hankalaa ymmärtää peliin alun perin suunniteltuja ”power up”- ja ”power down”-porttien merkityksiä. Pelin voittoehtoa oli myös hankala selittää pelaajille, ja se aiheutti väärinymmärryksiä. Tavoitteet sisäistettyään pelaajat kuitenkin pitivät pelistä ja sanoivat sen olevan mielenkiintoinen ja koukuttava.



Kuva 17. Ensimmäinen vertikaalinen viipale kehitetystä puzzlepelistä

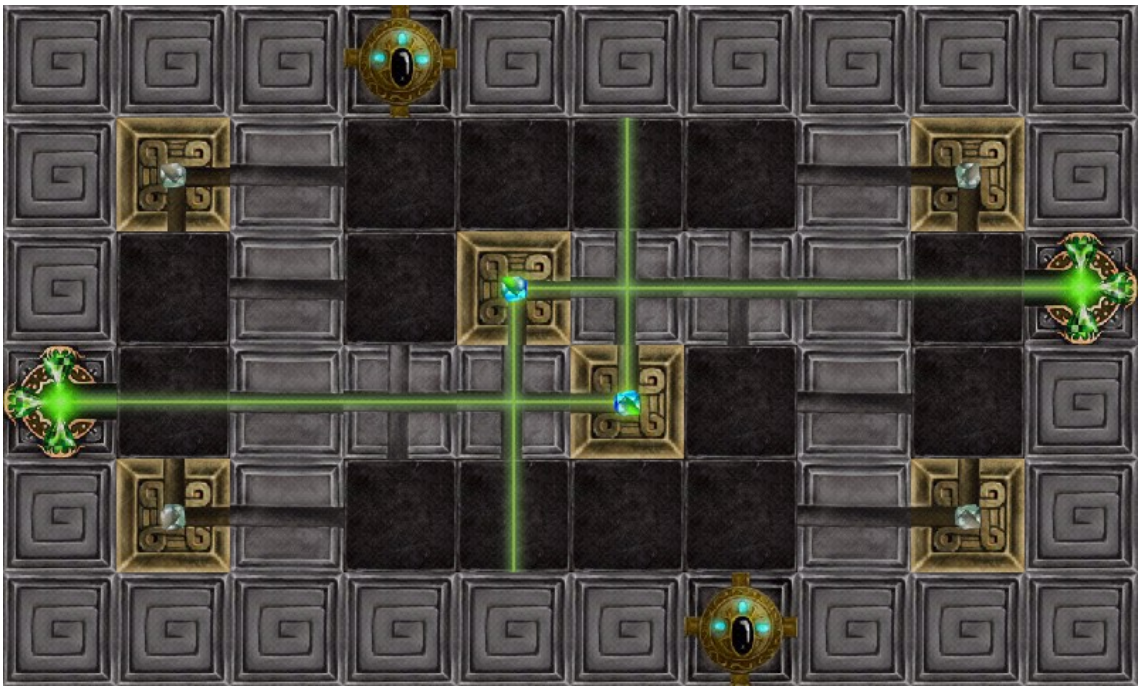
Alkuperäinen idea oli, että lasersäde tuli ohjata tiettyjen porttien läpi ennen kuin säde voidaan ohjata lopulliseen maaliin. Näiden porttien tarkoitus oli vaikeuttaa ratkaisun hahmottamista ja löytämistä. Porttien vuoksi pelaajan tavoite oli vaikeampi, eikä pelaaja voinut enää vain ohjata sädettä suoraan maaliin. Kun peliin lisättiin vihreitä portteja joista säteen tuli kulkea läpi, niin syntyi luonnollinen idea lisätä myös punaisia portteja joista säde ei saisi mennä läpi. Joten pelin tavoite oli seuraava: ”Ohjaa sädettä vihreiden porttien läpi saadaksesi energiaa ja vältä punaisia portteja joista menetät energiaa, kun sinulla on tarpeeksi energiaa, niin ohjaa säde koneeseen (maaliin) jolloin se käynnistyy”.

Toinen havaittu ongelma oli, että ruudut jäivät liian pieniksi alle neljätuumaisilla näytöillä. Ruutujen määrää tulisi vähentää, mutta tällöin porteille ei jäisi juuri tilaa. Seuraavassa versiossa oli tavoitteena poistaa hämmennystä aiheuttavat portit ja suurentaa ruutuja vähentämällä niiden kokonaismäärää.



Kuva 18. Puzzlepelin toinen verio ilman portteja

Toisen iteraation pelitestaamisessa havaittiin, että muutokset olivat menneet oikeaan suuntaan. Peli oli nyt selkeämpi ja opittavampi. Kentän komponentit olivat nyt suurempia ja niitä oli helpompi liikuttaa. Portit oli poistettu ja tilalle oli kehitetty kahdentyyppisiä ”on/off”-koneita. Toiset koneet menivät päälle, kun niihin ohjattiin lasersäde ja toiset koneet menivät pois päältä, jos niihin ohjattiin lasersäde. Pelin tavoite oli nyt siis seuraava: ”Ohjaa lasersäde koneisiin, jotka ovat OFF-tilassa, kunnes kaikki koneet ovat ON-tilassa voittaaksesi pelin.”. Testaajat ymmärsivät pelin tavoitteen huomattavasti nopeammin ja osa jopa ilman minkäänlaista ohjeistusta. ON/OFF-mekaniikka vaikutti kuitenkin vielä hieman vaikeasti ymmärrettävältä. Osa testaajista piti pystysuuntaisesta pelikentästä. Pystysuuntainen kenttä johti kuitenkin siihen, että osa pelaajista piti puhelinta huonomman käden kämmenellä ja pelasi peliä paremman käden etusormella. Tämä oli huomattavasti kömpelömpi ja hitaampi tapa pelata kuin vaakatasossa molempia peukaloita käyttäen. Pelin opittavuudessa ja käytettävyydessä oli siis vielä parantamisen varaa.



Kuva 19. Kolmas verio puzzlepelistä viimeistetyillä grafiikoilla

Kolmannesta versiosta päädyttiin poistamaan ”ON/OFF”-mekaniikka kokonaan, jolloin pelin tavoitteesta tuli huomattavasti yksinkertaisempi: ”Ohjaa säde jokaiseen maaliin”. Tavoite tuotiin esille myös jokaisessa pelinäkymässä havainnollistavilla kuvilla niin, että pelaaja oppisi ymmärtämään pelin tavoitteen huomaamattaan.

Lopullisten grafiikoiden suunnitteluun käytettiin huomattavasti aikaa, jolloin niiden tuottaminen ulkopuolisella tekijällä oli nopeampaa ja halvempaa. Suunnittelusta huolimatta grafiikoita jouduttiin iteroimaan myös useaan otteeseen. Grafiikoiden ensimmäisissä versioissa säde kulki erilaisten tunneleiden ja putkien läpi, mutta pelitestauksessa havaittiin, että säteen katoaminen näkyvistä vaikeutti monimutkaisempien kenttien hahmottamista. Lopulta päädyttiin käyttämään sellaisia värejä ja muotoja, joiden kanssa säde erottuu mahdollisimman hyvin muusta kentästä.

6 Yhteenveto

Insinöörityön ensimmäinen ja tärkein tavoite saavutettiin mallikkaasti eli uusi peli kehitettiin onnistuneesti ideasta julkaisuun. Oikeastaan opinnäytetyön aikana pelistä julkaistiin kaksi eri versiota kohtuullisella menestyksellä. Ensimmäistä ja ilmaista versiota (Dr. Laser) on ladattu jo yli 300 000 kertaa. Toisen version (Sampo Lock) oikeudet on myyty eteenpäin ja ainakin Google Playssa sitä on ladattu joitakin tuhansia

kertoja. Pelin tekeminen käynnisti nykyisen työurani, jonka johdosta insinöörityön kirjallinen osuus oli jäädä kesken, jolloin jouduin hieman karsimaan alkuperäistä insinöörityöaihettani. Alun perin halusin myös lyhyesti käsitellä pelin julkaisua ja markkinointia Google Playssa.

Pelisuunnitelussa ei otettu huomioon monetizaatiota osittain, koska pelin oli tarkoitus olla alunperin vain harjoitustyö. Myöhemmin mukaan tuli kuitenkin asiakas, joka tosin halusi tehdä pelistä vain promotuotteen ja mainostaa sillä tulevaa elokuvaansa. Mikäli monetizaatio olisi kuitenkin otettu huomioon pelisuunnittelussa alusta alkaen, niin pelistä olisi voitu tehdä itsessään kannattava tuote. Olisinkin halunnut sisällyttää työhöni tutkimuksen erilaisista monetizaatio-, ansainto- ja rahoitusmalleista. Jouduin kuitenkin jättämään näiden tutkimisen opinnäytetyön ulkopuolelle.

Projektin tekemisen myötä opin tuntemaan Androidin ohjelmistoalustana erittäin hyvin. Mikäli kuitenkin tekisin pelin uudestaan, niin käyttäisin suoraan jotain olemassa olevaa pelimoottoria kuten Unitya tai Coronaa. Tällöin peli olisi huomattavasti helpompi kääntää myös muille alustoille. Onneksi peli on kuitenkin kohtuullisen yksinkertainen, eikä sen kääntäminen tarvittaessa muille alustoille olisi mahdotonta.

Mikäli peliä vielä kehitetään, niin monetizaation lisäksi peli kaipaisi enemmän tehosteita ja animaatioita. Käytetyt graafiset elementit suunniteltiin niin, että pystyin itse tekemään kaikki tarvittavat grafiikat rajoittuneella osaamisellani. Lisäksi sisällöntuottamisen helpottamiseksi peli kaipaisi ehdottomasti pelaajalle tavan luoda omia kenttiä ja jakaa niitä ystäviensä kesken. Satunnaiskenttägeneraattorin tekeminen olisi myös mielenkiintoinen lisä peliin, vaikka sillä ei voida kokonaan korvata käsin suunniteltuja kenttiä.

Lähteet

1 Schell, Jesse. 2010. The Art of Game Desing: A Book of Lenses. Elsevier Inc.

2 Rogers, Scott. 2010. Level UP: The Guide To Great Video Game Design. John Wiley & Sons, Ltd.

3 Wikipedia, Leikki [verkkodokumentti] <http://fi.wikipedia.org/wiki/Leikki> Viitattu: 28.04.2014.

4 Wikipedia, Conway's Game of Life [verkkodokumentti] <http://en.wikipedia.org/wiki/Conway's_Game_of_Life> Viitattu: 29.04.2014.

5 Marc LeBlanc, Eight Kinds of Fun [verkkodokumentti] <http://8kindsoffun.com/> Viitattu: 29.04.2014.

6 Wikipedia, Maslow's hierarchy of needs [verkkodokumentti] <http://en.wikipedia.org/wiki/Maslow%27s_hierarchy_of_needs> Viitattu: 30.04.2014.

7 Anders Hejdenberg, The Psychology Behind Games [verkkodokumentti] <http://www.gamasutra.com/view/feature/2289/the_psychology_behind_games.php> Viitattu: 01.05.2014.

8 GitHub, Tiled Map Editor Wiki [verkkodokumentti] <https://github.com/bjorn/tiled/wiki> Viitattu: 02.05.2014.

9 Mike Wiering, Tile Studio [verkkodokumentti] <http://tilestudio.sourceforge.net/> Viitattu: 02.05.2014.

10 CodePlex, tIDE Documentation [verkkodokumentti] <http://tide.codeplex.com/documentation> Viitattu: 02.05.2014.

11 Balsamiq, Balsamiq Mockups [verkkodokumentti] <http://balsamiq.com/> Viitattu: 03.05.2014.

12 Android API Reference, XmlPullParser [verkkodokumentti] <http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html> Viitattu: 04.05.2014.

13 Android Desing, UI Overview [verkkodokumentti] <http://developer.android.com/design/get-started/ui-overview.html> Viitattu: 05.05.2014.

14 Wikipedia, Vertical slice [verkkodokumentti] <http://en.wikipedia.org/wiki/Vertical_slice> Viitattu: 06.05.2014.

15 NDFC Helsinki, Yritys kuvaus [verkkodokumentti] <http://www.auroranord.org/> Viitattu: 07.05.2014.

16 Sven Koenig, Fast Replanning for Navigation in Unknown Terrain [pdf-verkkodokumentti] <http://pub1.willowgarage.com/~konolige/cs225b/dlite_tro05.pdf> Viitattu: 08.05.2014.

17 Wikipedia, D* [verkkodokumentti] <http://en.wikipedia.org/wiki/D*> Viitattu: 08.05.2014.

18 Android Api Guide, Graphics Overview [verkkodokumentti]
<https://developer.android.com/guide/topics/graphics/overview.html> Viitattu: 09.05.2014.

19 Android Api Guide, Layouts [verkkodokumentti]
<http://developer.android.com/guide/topics/ui/declaring-layout.html> Viitattu: 09.05.2014.

20 PopCap Games, Mobile Gaming Research [pdf-verkkodokumentti]
<http://www.infosolutionsgroup.com/popcapmobile2012.pdf> Viitattu: 10.05.2014.